CLAIMS:

1. A method for updating program code stored in a memory, which memory comprises a plurality of memory sectors, the method comprising

- transforming an updated input code into an updated program code version to be stored in a memory, which memory has stored thereon a current program code version occupying a first set of the memory sectors of the memory, wherein the updated program code version occupies a second set of memory sectors when stored in the memory;

characterised in that the transforming further comprises

- receiving a representation of the current program code version; and
- performing at least one optimisation step adapted to decrease the number of memory sectors of the second set of memory sectors occupied by the updated code version that are different from the corresponding memory sectors of the first set of memory sectors occupied by the current program code version.

2. A method according to claim 1, wherein the representation of the current program code version comprises at least one of a current image of the first set of memory sectors and a map file description of the current image of the first set of memory sectors.

3. A method according to claim 1 or 2, wherein the input code comprises a number of object modules; and wherein the transforming comprises linking the number of object modules.

4. A method according to claim 1 or 2, wherein the input code comprises at least one source code module; wherein the transforming comprises

- compiling the at least one source code module resulting in a number of object modules; and
- linking the number of object modules;

and wherein performing at least one optimisation step comprises

- generating feedback data during the linking step; and
- re-compiling at least a subset of the source code modules based on the feedback data and resulting in a number of modified object modules; and
- performing the linking step based on the number of modified object modules.

5. A method according to claim 3 or 4, wherein the optimisation step comprises determining a layout of said object modules in memory.

6. A method according to claim 5, wherein determining the layout of said object modules in memory comprises

- detecting an updated first object module having a different size than a corresponding first current object module, and an updated second object module equal to a corresponding second current object module, which updated second object module has a base address larger than the base address of the updated first object module; and
- padding the detected updated first object module with a predetermined memory content of a predetermined padding size resulting in a padded updated first object module; wherein the padding size is selected to cause the base address of the updated second object module to be equal to the base address of the corresponding second current object module.

7. A method according to claim 5 or 6, wherein determining the layout of said object modules in memory comprises

- detecting an updated first object module that is larger than a corresponding first current object module;

- moving a predetermined part of the updated first object module to a different memory sector resulting in a reduced updated first object module and a moved part of the updated first object module;
- inserting a relay to the moved part of the updated first object module in the reduced first updated memory sector.

8. A method according to any one of claims 1 through 7, wherein the transforming further comprises controlling the optimisation step by at least one optimisation parameter.

9. A method according to claim 8, wherein the at least one optimisation parameter includes a parameter determining a maximum allowed increase in size caused by the optimisation step.

10. A method according to claim 8 or 9, wherein the at least one optimisation parameter includes a parameter determining a maximum allowed number of relays introduced by the optimisation step.

11. A method according to any one of claims 1 through 10, further comprising generating a delta file representative of differences between the current program code version and the updated program code version.

12. A method according to any one of claims 1 through 11, wherein the memory is a flash memory.

13. A data processing system for updating program code stored in a memory, the memory comprising a plurality of memory sectors, the data processing system being suitably programmed to perform the steps of the method according to any one of claims 1 through 12.

14. A computer program product comprising program code means adapted to cause a data processing system to perform the method according to any one of claims 1 through 12, when said program code means are executed on the data processing system.

15. A computer program product according to claim 14, wherein the computer program product comprises a linker module.

16. Use of a method according to claim 1 through 12 for the reprogramming of portable radio communications equipment.